2019 IEEE World Congress on Services (SERVICES)

Data Encryption and Fragmentation in Autonomous Vehicles using Raspberry Pi 3

Sahand Murad*, Asiya Khan*, Stavros Shiaeles* and Giovanni Masala†
*University of Plymouth, UK
e-mail: sahand.murad;;asiya.khan;stavros.shiaeles@plymouth.ac.uk

†Manchester Metropolitan University, UK
e-mail: g.masala@mmu.ac.uk

Abstract—**Autonomous vehicles have huge potential in improving road safety and congestion. Towards the road map of full autonomy, each vehicle will be able to communicate with other vehicles within the network of vehicles to improve congestion and notify emergencies. Many architectures for communication between vehicles are centralised, typically using cloud servers. The security and trust of that communication is paramount. Therefore, the aim of this paper is to propose a novel method for encrypting and fragmenting data in various cloud providers in order to protect the anonymity and increase the uncertainty for an attacker having access to the data on cloud. Our experimental results seem promising and we were able to achieve good results with low overhead in transmission.**

Keywords-component; data fragmentation, encryption, autonomous vehicles

I. INTRODUCTION

Autonomous cars are intelligent systems, which are able to do physical tasks without human interaction. Autonomous vehicles are used in industrial environment, transport and military. These vehicles are increasingly becoming intelligent agents that have the capability to learn from their environment. Autonomous cars have several sensors with connectivity between them. Most cars manufactured now use some autonomous features such as lane keeping, Adaptive Cruise Control (ACC) and automatic parking. In order to research on such vehicles, Induct Technology created Navia, which is a robotically driven electric shuttle, with a maximum speed of 20 Km/h. The shuttle has an optical camera and four Lidar sensors on it, and has been tested in many universities in England, Singapore and Switzerland [1]. In each car, it is possible to install many kinds of sensors e.g. LIDAR, GPS, Camera sensor, which are able to perform specialized task.

In the last decade, driverless vehicles have been piloted on the roads. In addition, advanced driver assistance systems and autonomous vehicles continue to increase rapidly, e.g. the 2004 and 2005 DARPA challenges for vehicles to autonomously navigate via desert terrain. Further, the DARPA challenge in 2007 developed and tested cars that independently explored via a mock urban condition amid traffic, have created significant excitement and research enthusiasm for the field of autonomous driving [2] [3].

The increased connectivity and interaction of an autonomous vehicle gives rise to vulnerabilities in data integrity. A single vehicle compromised can lead to security hazards affecting multiple vehicles. Hence, it is important to design robust systems that ensure data integrity and are very difficult to be hacked. Data encryption offers some security. Therefore, to make data secure from hackers it is better to encrypt data before sending data to the cloud to protect the data from the attackers; data is encrypted using a key verified by the identity of the sender [4]. There are many ways to protect data, such as hiding identities of communication parties; however, still let them authenticate each other, encryption and authentication [5]. Moreover, as managing and protecting key is a problem, authors in [6] propose a cloud stash, which splits the files to multi shares, and send it to multiple clouds where the files are reconstructed by the threshold. Whereas, authors in [7] propose a threshold scheme to divide the data in a number of pieces in such a way that the individual pieces do not reveal any information about the data. Another way to keep the data safe from hackers is symmetric and asymmetric encryption, in symmetric encryption sender and receiver use the same key to encrypt and decrypt data but in asymmetric each user has their own private key to encrypt and decrypt the data [8].

Therefore, the main contribution of this paper is to propose a scheme to protect the anonymity of the vehicular data, thus, increasing the uncertainty for an attacker to access the data on the cloud. Vehicular data is collected from an autonomous vehicle with ultrasound sensors and camera on a Raspberry Pi 3 microcontroller [9]. We collect both text and video data and propose a novel way to achieve data integrity. We first split the data (both text and video) into smaller chunk sizes of 20KB, 15KB, 10KB and 5KB, and encrypted it. We then compared that to the whole file encrypted in terms of performance metrics of time, CPU usage and size of files. We used Python 3 for programming and for encryption we used Pycrypto Aes CBC256.

The rest of the paper is organized as follows. Section II presents the literature review. In section III we present the proposed method, whereas, the results are presented in Section IV. Section V concludes the paper and presents areas of future work.

978-1-7281-3851-0/19/$31.00 ©2019 IEEE
DOI 10.1109/SERVICES.2019.00058

II. LITERATURE REVIEW

There have been several ways in which data protection from hackers has been presented in literature. Encryption and fragmentation techniques offer secure ways to protect data from hackers. Researchers in [6] proposed a cloud stash to split data to multi shares, where threshold shares are required to reconstruct the file. They used multithreading to manage secret sharing in to multiple clouds. Therefore, when the client tries to download data and it is corrupted, cloud stash can reconstruct a new one from another cloud. To protect and make data more secure from attackers, work presented in [10] encrypted the data before sending it to the cloud to make it more secure. Only the clients had access to the decryption keys making it is difficult for the attackers to get the data. The work presented in [11] [15]proposed a new technique to fragment the original file to chunks, they used random pattern fragmentation for splitting data, each split file had the same length as the associated pattern. Moreover, the aim of using this method is to protect the data from attackers, as the attackers do not know the information about the length of each split data, therefore, they cannot get the data. Authors in [15] present a new approach for data fragmentation combined with AES 256 CBC to encrypt the data and split to multiple chunks, and the chunks were sorted to split files by using a random pattern fragmentation. In [4] authors propose a taxonomy of defence against attacks on autonomous vehicles to enable targeted defence to be developed. According to [12] to make the data more secure they proposed to split the data, then after splitting the data, the header was stored in another file of smaller size, they then added some bytes to the header making it harder to be attacked as it appears to be like other header files. This will make it harder for the attacker to find the header because when attacker attacks the data inside the cloud, they cannot find the header because all the split files have the same size, as it shows that the header was not sent to the cloud, so the attackers cannot defragment the split data without headers. Moreover, to make data more safe from eyes of the attackers, authors in [13] proposed a way to encrypt data by using master key and symmetric encryption. For each authorized client key encryption key (KEK) obtained via passphrase and also by using KEK the master key has been encrypted, after the data encrypted each client the master key and client identifier encrypted depending on user key encryption key. For decrypting the data, the user enters a passphrase to the client which obtains the KEK, so the client decrypts the master key and data. From the literature, we conclude that many researchers have worked on autonomous cars data splitting data or encrypting data and splitting headers, but no one has tried to split and encrypt the headers of different files.

III. PROPOSED METHOD

The work presented in this paper focuses on splitting and encrypting files to smaller chunks, encrypt the header and send those chunks to the multiple providers. In each encrypted header there is information about the location of next data which helps the client locate the data to reconstruct it easily, so by using the proposed method the attackers cannot reconstruct the data as they cannot understand anything from the data.

Fig. 1 shows the proposed scheme for vehicular data security. The right side of Fig. 1 shows the block diagram of the autonomous vehicle using the Raspberry Pi microcontroller. Both text and video data collected from the vehicle is then split into multiple chunks using Python 3 (left side of Fig. 1). To hide the data from the attacker's eyes and make it more private, we proposed to split data to smaller chunks and encrypt the headers. We use the AES encryption method. The data is then re-joined and send to multiple clouds thus making the data more secure as described in Fig. 1.

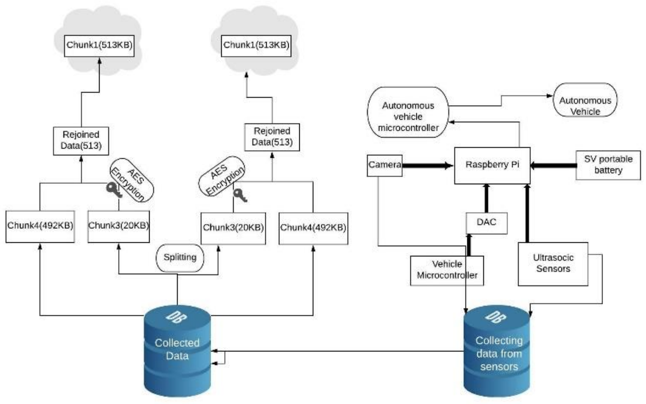

Fig. 1. Block diagram of the autonomous vehicle and Data Collection.

This section outlines the experimental methodology, conversion of the mobility vehicle to be fully autonomous with sensors connected and the proposed data encryption and fragmentation scheme.

A. Experimental Methodology

The vehicle used for this project is a Capricorn Electric Wheelchair from Better life Healthcare [14] as shown in Fig. 2a. It is a small, four wheeled vehicles with caster type front wheels, two fixed driven rear wheels and powered by two 12V batteries. It is driven by two separate electric motors, which are connected directly to each of the rear wheels. It has a maximum speed of 4mph, a maximum incline of 6° and a turning circle of radius 475mm. The maximum range of the wheelchair is 9.5km. The tyres are solid and have a larger radius than many other models of its type, helping to improve performance on rough or uneven surfaces.

This section will present the conversion of the mobility scooter into an autonomous vehicle controlled by Raspberry Pi 3. It will further describe the connection of ultrasonic sensors and camera.

B. Connecting the Raspberry Pi 3

The autonomous vehicle was built from a mobility scooter as shown in Fig. 2a. The scooter had an inbuilt microcontroller shown in Fig. 2b which was used as a communicative tool between the Raspberry Pi version 3 and the vehicle's motors.

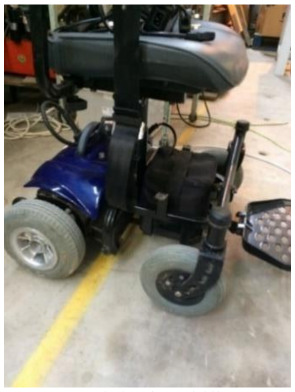

Fig. 2a. Original mobility scooter

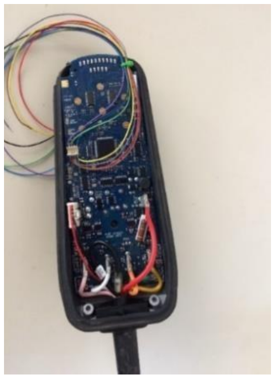

Fig. 2b. Autonomous vehicle microcontroller

In order to make space for a platform on which the system can be installed, the chair was removed, as was the housing surrounding the frame of the vehicle. The central column between the chair and the frame was also removed, allowing the new chassis to be placed over the frame. The newchassis is shown in Fig. 3, whereas, the block diagram is presented in Fig. 1 showing the connections of the Raspberry Pi with the sensors and the vehicle's controller. The chassis shown in Fig. 2a has enough space for the control panel – rewired to connect the Raspberry Pi directly to the joystick input, the Pi itself, and two breadboards with which the circuitry could be modified during the built and testing process. The chassis is designed so additional components and sensors can be added. Two digital to analogue converters were installed, controlling both forwards/backwards motion and the yaw of the vehicle, respectively. The front wheels were fixed in place by the removal of the bearings contained in the shafts. This allowed the connecting bolts to be tightened fully and restricting the motion of the vehicle to forwards and backwards.

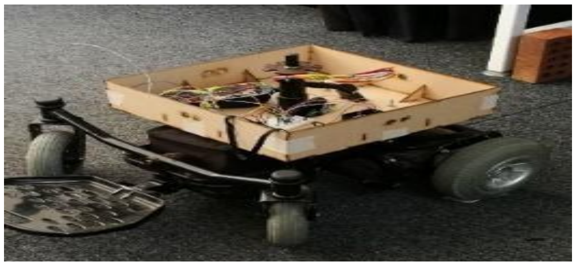

Fig. 3. The autonomous vehicle modified from the mobility scooter

For the vehicle to be autonomous it would have to be controlled by the General-Purpose Input Output (GPIO) pins on the Pi which would send signals emulating the joystick. The GPIO pins work with digital signals therefore a Digital to Analogue Converter (DAC) (Fig. 2b) would be required to alter the signal type. An Adafruit MCP4725 DAC [20] was used and functioned well with the Raspberry Pi.

To ensure the vehicle was mobile, a portable battery producing 5V was powering the Pi and a laser-cut housing was designed to hold the system in place and absorb forces from potential collisions (Fig.3).

C. *Connecting Ultrasonic Sensors to the Raspberry Pi 3*

Ultrasonic sensors provide basic object-detection autonomy to the vehicle. The HC-SR04 sensor was used which could work with the Pi's GPIO pins through jumper wires. The principle of the HC-SR04 is that there are four pins: power, trigger, echo, and ground. The power and ground were connected directly to the Pi's voltage and ground pins, the trigger acts as a 'starting gun' for the sensor signifying when to produce a soundwave, and the echo receives the soundwave. While these are binary input/output functions they can be used on Python to determine the distance of the closest object. The programming logic is shown in in Fig. 1**.** In order to connect the ultrasonic sensors to the Raspberry Pi, the circuit was adjusted so that all of the triggers were controlled by the same i/o pin on the Pi, keeping the amount of i/o pins used to a minimum of x+1, where x is the number of sensors used in the design. Each of the sensors outputs the result to an array which is continually updated, and it is this array which can be communicated to an infrastructure hub. As sensors are connected to the Raspberry Pi, and the microcontroller makes a connection between vehicle and Raspberry Pi, we collected data from the sensors (text data) and video data from the camera. This process has been described earlier in Fig.1.

D. DATA ENCRYPTION AND FRAGMENTATION

Text data from the ultrasonic sensors and video data from the camera was collected from the autonomous vehicle. We used AES encryption algorithm as it is the most widely used encryption method, it is symmetric encryption which uses one key for encryption and decryption. We collected the data from the autonomous vehicle ultrasound sensors and from the camera. We then encrypted the whole text and video file, then we fragmented both the text and video files in to four fragments of sizes 5KB, 10KB, 15KB and 20KB. The size of the whole video file was 512KB and the text file was 247KB. This allowed us to make comparison in terms of time, size and CPU usage for the full-size file and when fragmented. The full-size video and data files were encrypted with AES method. Finally, the fragmented video and text files were encrypted with the AES method as described earlier in Fig.1 After fragmenting the data files in 4 different sizes, we encrypted them and then re-joined it with the remaining files i.e. 507, 502, 497 and 492KB, and send it to multiple clouds, so when the attacker tries to attack the data in cloud they will not be able to decrypt it. This is because the hacker cannot understand anything from the data because it will be difficult to find all the chunks and determine the difference between header and body. For reconstructing data, we used the same method i.e. the client downloads the data chunks and decrypt the headers, then re-joining the chunks and the client has access to the original data. After the client has downloaded text or video data, As each client has his own key so it will be easier for the client to get first chunk after that they can find all chunks as each chunk has the information about previous and next chunk.

IV. RESULTS AND DISCUSSIONS

Once the data was collected from the autonomous vehicle, we have analysed the results in terms of the comparison in time, size and CPU utilization of the text and video files. The data files were encrypted as the whole file and as the fragmented chunks of size 5, 10, 15 and 20KB. The size of the whole video file is 512KB and of the text file is 247KB.

We found that encrypting whole file needs more time and storage rather than encrypting first part of the file. There is change in the size of files before encrypting and after encrypting first 20, 15, 10 and 5 KB. The size of both files (text and Video) increased after encrypting to 20.2KB. For both (encrypting part of file then re-joining and encrypting whole file), as the size of file before encrypting was 20KB or other sizes and after encrypting increased, encrypting whole of the text file results to bigger size of file rather than encrypting just some KB of file as the file size was 247 (253943bytes) but after encrypting the size increased to 248 (254247 bytes).

Table I. Time and CPU utilization for splitting different sizes of video and text file

Text / Video	Time	CPU usage
20 KB	0.008 / 0.017	%2.15 / %3.88
15 KB	0.007 / 0.015	%1.85 / %3.6
10 KB	0.006 / 0.013	%1.65 / %2.13
5 KB	0.005 / 0.011	%1.4 / %1.77

Table 1 shows that splitting 512 KB to 20 KB and 492KB of both files take more time than splitting the same size to 15 KB and 497KB, 10 KB and 502KB, and 5 KB and 507 KB, also splitting 15 KB of video file takes more time rather than splitting the same size of text file. The chart shows that splitting 10 KB takes more time compared to the time to split 5 KB. In addition, splitting 20 KB takes more time than 15KB, 15KB takes more time for splitting than 10 KB and 5 Kb of video file is also for the text file splitting 20 KB takes more time than splitting other sizes of the same file. Moreover, from Table 1 we can see that splitting 20 KB, 15 KB, 10 KB and 5 KB of video file takes more time rather than splitting same sizes of text file. Table 1 also shows the CPU utilization for splitting text and video files in different sizes, from the chart we can see that splitting 20 KB need more CPU usage than other sizes of video and text files, also for splitting 15 KB CPU utilization is more than 10 Kb and 5 KB for both files. On the other hand, splitting 20KB of video file needs more CPU usage compared to 12 KB of video rather splitting 20KB of text file.

We can see from Fig. 4 the time to encrypt 20 KB of video file needs more time rather than encrypting same size of text file, at same time encrypting 20 KB takes more time than other sizes. Encrypting 15 KB of video and text file needs more time than encrypting 10 KB of both files, but in encrypting 20 KB of video file takes more time rather than encrypting same size of text file. Encrypting 5 KB of text takes less time than encrypting same files of size 20KB, 15 KB, and 10 KB and less than same size of video file. Moreover, the chart represents that encrypting 512 KB of video and text file takes more time than encrypting just 20, 15, 10 and 5 KB.

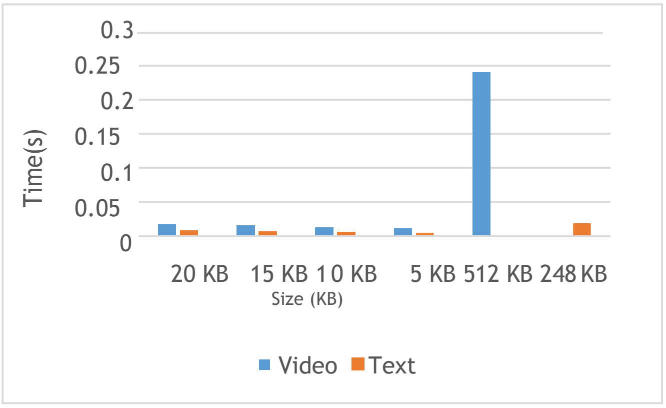

Fig. 4 Time for encrypting different sizes of video and text file

Fig. 5shows that usage of CPU for encrypting 20KB of video is higher than encrypting same size of text file, compared to other sizes. CPU usage of 20 KB encryption is higher than encrypting other sizes. Moreover, when we encrypted 15KB of video file the CPU usage is higher than encrypting 15KB of text file also when we encrypted 10KB for both video and text we can see that the CPU usage of encrypting 10 KB of video file is bigger than encrypting the same size of text file. Also the CPU usage for encrypting 5KB of text file is less than the video file. Overall from Fig. 5 we can see that encrypting 5KB results in less CPU usage compared to other sizes.

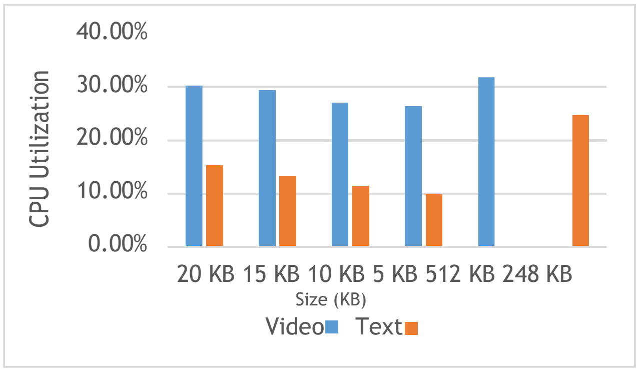

Fig. 5 CPU utilization for encrypting different sizes of video and text file

From the experimental results, we can see that the best way to save time and CPU usage is to split and encrypt 5KB of 512 KB rather than other sizes or encrypting the whole file without splitting.

V. Conclusion and future work

In this paper, our experimental results shown that splitting and encrypting 5 KB is the ideal size as it is not using many CPU resources and memory compared to other sizes. We compared our results in terms of time, CPU utilization and size. This allowed us to determine which one is most useful method to make the data secure in the cloud and hence make it difficult for a hacker to reconstruct data. Splitting and encrypting different size of video and text file or encrypting whole file shows that less time, CPU usage and size is taken in splitting and encrypting 5KB rather than other sizes or encrypting the whole file, so it saves CPU utilization, time and storage. The privacy of data is at a higher level preventing a hacker to access the data as it is shared in multiple clouds and header of each chunk is encrypted.

In our future work, we will reconstruct data from multiple clouds with each chunk having its location also each chunk knows about next and previous chunk location which will help the client to reconstruct the data easily and decrypt data then send it back to the autonomous vehicle.

Acknowledgments

The authors would like to thank Mr Charlie Day, My Liam McEachen, Mr Stuart MacVeigh, Mr John Welsh and Dr Toby Whitley for their support in building the autonomous vehicle and general electronics.

References

[1] Zhang, R. and M. Pavone, Control of robotic mobility-on-demand systems: a queueing-theoretical perspective. The International Journal of Robotics Research, 2016. 35(1-3): p. 186-203.

[2] Choi, W.-S., et al. Fast iterative closest point framework for 3D LIDAR data in intelligent vehicle. in 2012 IEEE Intelligent Vehicles Symposium. 2012. IEEE.

[3] Levinson, J. and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. in 2010 IEEE International Conference on Robotics and Automation. 2010. IEEE.

[4] Thing, V.L. and J. Wu. Autonomous vehicle security: A taxonomy of attacks and defences. in 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). 2016. IEEE.

[5] Chaum, D.L., Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 1981. 24(2): p. 84-90.

[6] Alsolami, F. and T.E. Boult. CloudStash: using secret-sharing scheme to secure data, not keys, in multi-clouds. in 2014 11th International Conference on Information Technology: New Generations. 2014. IEEE.

[7] Shamir, A., How to share a secret. Communications of the ACM, 1979. 22(11): p. 612-613.

[8] Esslinger, B., CrypTool. Available via www. cryptool. de, 2008.

[9] Day, C., McEachen, L., Khan, A., Sharma, S. and Masala, G., L. Pedestrian Recognition and Obstacle Avoidance for Autonomous Vehicles using Raspberry Pi. accepted for presentation at the Intelligent Systems Conference (IntelliSys) 2019, 5-6 September 2019 in London, United Kingdom.

[10] di Vimercati, S.D.C., et al., Encryption and fragmentation for data confidentiality in the cloud, in Foundations of security analysis and design VII. 2013, Springer. p. 212-243.

[11] Santos, N. and G.L. Masala. Big Data Security on Cloud Servers Using Data Fragmentation Technique and NoSQL Database. in International Conference on Intelligent Interactive Multimedia Systems and Services. 2018. Springer.

[12] Bahrami, M. and M. Singhal. A light-weight permutation based method for data privacy in mobile cloud computing. in 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering. 2015. IEEE.

[13] Jonas, P.E., A.L. Roginsky, and N. Zunic, *Encrypting data for access by multiple users*. 2009, Google Patents.

[14] Betterlife Healthcare, 2018. Betterlife Capricorn Electric Wheelchair.

[15] N. Santos, S. Lentini, E. Grosso, B. Ghita and G. Masala, "Performance Analysis of Data Fragmentation Techniques on a Cloud Server" in press on the, International Journal of Grid and Utility Computing, InderScience publishers.